cofc

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Starr, et al.		Patent No.:	6,996,070
Issued:	September 6, 2005	Ser. No:	10/729,111
Filing Date:	December 5, 2003	Examiner:	Robert Wilson
Atty. Docket No:	ALA-026	GAU:	2661

For: TCP/IP OFFLOAD DEVICE WITH REDUCED SEQUENTIAL PROCESSING

February 13, 2006

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Request for Certificate of Correction under 37 C.F.R. §1.322

Sir:

Enclosed are three copies of a Certificate of Correction for the above-referenced patent. Also enclosed is a copy of an Amendment, which was filed August 29, 2006, and which shows the mistakes appear to have been made by the Patent Office, and so no fee is required. Please issue the enclosed Certificate of Correction.

Respectfully submitted,

Mark Lauer
Reg. No. 36,578
6601 Koll Center Parkway
Suite 245
Pleasanton, CA 94566
Tel: (925) 484-9295
Fax: (925) 484-9291

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313, on February 13, 2006.

Date: 2-13-06

Mark Lauer

PTO/SB/44 (04-05)
Approved for use through 04/30/2007. OMB 0651-0033
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.
(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 6,996,070

APPLICATION NO.: 10/729,111

ISSUE DATE : February 8, 2006

INVENTOR(S) : Starr et al.

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 34, line 20, correct the following word in claim 10:

"racket" should be "packet"

MAILING ADDRESS OF SENDER (Please do not use customer number below):
Mark Lauer
6601 Koll Center Parkway, Suite 245
Pleasanton, CA 94566

FEB

PTO/SB/44 (04-05)
Approved for use through 04/30/2007. OMB 0651-0033
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,996,070

APPLICATION NO.: 10/729,111

ISSUE DATE : February 8, 2006

INVENTOR(S) : Starr et al.

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 34, line 20, correct the following word in claim 10:

"racket" should be "packet"

MAILING ADDRESS OF SENDER (Please do not use customer number below):
Mark Lauer
6601 Koll Center Parkway, Suite 245
Pleasanton, CA 94566

FEB 23 2006

PTO/SB/44 (04-05)
Approved for use through 04/30/2007. OMB 0651-0033
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.
(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 6,996,070

APPLICATION NO.: 10/729,111

ISSUE DATE : February 8, 2006

INVENTOR(S) : Starr et al.

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 34, line 20, correct the following word in claim 10:

"racket" should be "packet"

MAILING ADDRESS OF SENDER (Please do not use customer number below):
Mark Lauer
6601 Koll Center Parkway, Suite 245
Pleasanton, CA 94566

FEB 22 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Starr, et al.

Assignee: Alacritech, Inc.

Title: "TCP/IP Offload Device With Reduced Sequential Processing"

Appl. No.: 10/729,111 Filing Date: December 5, 2003

Examiner: Robert Wilson Art Unit: 2661

Docket No.: ALA-026

August 29, 2005

Mail Stop Amendment
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT

Dear Sir:

In response to the Office Action dated June 7, 2005, the Applicants respond as follows.

Amendments to the Specification are set forth starting on page 2 of this Amendment.

Amendments to the Claims are reflected in the listing of claims that begins on page 4 of this Amendment.

The **Remarks** begin on page 17 of this Amendment.

IN THE SPECIFICATION

Please amend paragraph [0003] as follows:

[0003] USP 6,247,173 describes ~~on~~ <u>one</u> example of a TOE. The TOE device includes a processor as well as several other devices. The processor on the TOE executes firmware instructions that are stored on the TOE device. As networking speeds have increased, so too have the processing demands imposed on the processor of such a TOE. One way TOE processing power has been increased is by increasing the clock rate of the processor. This increases the rate at which the processor fetches and/or executes instructions. There are, however, practical limits on how high the processor clock rate can be increased. Advances in semiconductor processing technology over time have allowed ever increasing processor clock rates, but it is envisioned that the rate of increase will not be adequate to keep pace with the future demands on processing power due to even more rapid increases in network speeds.

Please amend paragraph [0032] as follows:

[0032] System 1 includes a host 3 and a network interface device (NID) 4. Host 3 may, for example, be embodied on a motherboard. NID 4 may, for example, be an expansion card that couples to the motherboard. Host 3 includes a central processing unit (CPU) 5 or CPU chip-set, and an amount of storage 6. In the illustrated example, storage 6 includes a combination of semiconductor memory and magnetic disc storage. CPU 5 executes software stored in storage 6. The software includes a network protocol processing stack including a media access protocol processing layer, an IP protocol processing layer, a TCP protocol

processing layer, and an application layer. The protocol layer on top of the TCP protocol processing layer is sometimes called a session layer and is sometimes called an application layer. In the description below, the layer ~~s~~ is referred to as the application layer.

Listing of Claims

1. (Original) A TCP offload engine (TOE) capable of offloading, from a host, TCP protocol processing tasks that are associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

combinatorial logic that receives said at least two TCP state values and said at least two packet header values and generates therefrom a flush detect signal indicative of whether an error condition has occurred, the two TCP state values and the two packet header values all being supplied to the combinatorial logic simultaneously.

2. (Presently Amended) <u>A TCP offload engine (TOE) capable of offloading, from a host, TCP protocol processing tasks that are associated with a TCP connection; the TOE comprising:</u>

<u>a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence</u>

FEB 23 2005

limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

combinatorial logic that receives said at least two TCP state values and said at least two packet header values and generates therefrom a flush detect signal indicative of whether an error condition has occurred, the two TCP state values and the two packet header values all being supplied to the combinatorial logic simultaneously ~~The TOE of Claim 1~~ , wherein the TOE transfers the receive packet sequence limit, the expected receive packet sequence number, the transmit sequence limit, the transmit acknowledge number, and the transmit sequence number to the host if the flush detect signal is indicative of the error condition.

3. (Presently Amended) The TOE of Claim ~~1~~ 5, wherein more than two values of the group of TCP state values are supplied simultaneously to the combinatorial logic, and wherein more than two values of the group of TCP state values are used to generate the flush detect signal.

4. (Presently Amended) The TOE of Claim ~~1~~ 5 , wherein more than two values of the group of packet header values are supplied simultaneously to the combinatorial logic, and wherein more than two values of the group of packet header values are used to generate the flush detect signal.

Applicants: Starr et al.
Serial No.: 10/729,111
Filing Date: December 5, 2003
Docket No.: ALA-026

5. (Presently Amended) A TCP offload engine (TOE) capable of offloading, from a host, TCP protocol processing tasks that are associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

combinatorial logic that receives said at least two TCP state values and said at least two packet header values and generates therefrom a flush detect signal indicative of whether an error condition has occurred, the two TCP state values and the two packet header values all being supplied to the combinatorial logic simultaneously ~~The TOE of Claim 1~~ , wherein the TOE can control a plurality of TCP connections, each of said TCP connections being associated with a TCB identifier, and wherein the first memory comprises a plurality of transaction control blocks (TCB), wherein the first memory can be addressed by a TCB identifier to access a TCB that contains TCP state information for a TCP connection associated with the TCB identifier.

6. (Presently Amended) The TOE of Claim ~~1~~ 2 , wherein the combinatorial logic is part of a state machine, the state machine being clocked by a clock signal,. wherein the combinatorial logic generates said flush detect signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal.

7. (Original) The TOE of Claim 6, wherein the state machine does not fetch instructions, decode the instructions, and execute the instructions.

8. (Original) The TOE of Claim 7, wherien the state machine causes the expected packet receive sequence number and the receive packet sequence limit values in the first memory to be updated in a single period of the clock signal.

9. (Presently Amended) The TOE of Claim ~~8~~ 2 , wherein the TCP connection was set up by a stack executing on the host, and wherein control of the TCP connection was then passed from the host to the TOE.

10. (Presently Amended) A TCP offload engine (TOE) capable of offloading, from a host, TCP protocol processing tasks that are associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

combinatorial logic that receives said at least two TCP state values and said at least two packet header values and generates therefrom a flush detect signal indicative of whether an error condition has occurred, the two TCP state

values and the two packet header values all being supplied to the combinatorial logic simultaneously, wherein the combinatorial logic is part of a state machine, the state machine being clocked by a clock signal, wherein the combinatorial logic generates said flush detect signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal, wherein the state machine does not fetch instructions, decode the instructions, and execute the instructions, and

~~The TOE of Claim 7~~ , wherein a packet having a payload is received onto the TOE, the TOE further comprising:

a DMA controller that moves the payload from the TOE to the host using a source address value, a size value indicating an amount of information to move, and a destination address value, wherein the state machine causes the source address value, the size value, and the destination address value to be supplied to the DMA controller in a single state of the state machine.

11. (Original) A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

a hardware state machine that receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the hardware state machine simultaneously, wherein the hardware state machine is clocked by a clock signal, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal.

12. (Presently Amended) <u>A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:</u>

<u>a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;</u>

<u>a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and</u>

<u>a hardware state machine that receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the hardware state machine simultaneously, wherein the hardware state machine is clocked by a clock signal, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values within</u>

approximately one period of the clock signal ~~The TOE of Claim 11~~, wherein the hardware state machine causes the expected packet receive sequence number and the receive packet sequence limit values in the first memory to be updated in a single period of the clock signal.

13. (Original) The TOE of Claim 12, wherein the expected packet receive sequence number and the receive packet sequence limit values are updated by simultaneously writing the expected packet receive sequence number value and the receive packet sequence limit value into the first memory.

14. (Presently Amended) The TOE of Claim ~~11~~ 12 , wherein the exception condition is a condition which results in control of the TCP connection being passed from the TOE to the host.

15. (Presently Amended) A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

a hardware state machine that receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal

indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the hardware state machine simultaneously, wherein the hardware state machine is clocked by a clock signal, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal ~~The TOE of Claim 11~~, wherein the first memory is a dual-port memory that has a first interface and a second interface, the hardware state machine reading from and writing to the first memory via the first interface, and wherein information passes from the host and into the first memory via the second interface.

16. (Original) The TOE of Claim 15, wherein the first memory comprises a plurality of TCB buffers, wherein one of the TCB buffers is associated with the TCP connection, and wherein a memory descriptor list entry associated with said TCP connection is stored in said TCB buffer associated with said TCP connection.

17. (Presently Amended) A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group

consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value;

a hardware state machine that receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the hardware state machine simultaneously, wherein the hardware state machine is clocked by a clock signal, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal; and ~~The TOE of Claim 11, further comprising:~~

a third memory that stores a plurality of socket descriptors, wherein one of the socket descriptors is associated with the TCP connection.

18. (Presently Amended) A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection, wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value;

a hardware state machine that receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal

indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the hardware state machine simultaneously, wherein the hardware state machine is clocked by a clock signal, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values within approximately one period of the clock signal ~~The TOE of Claim 11~~, wherein the second memory outputs a parse value along with said at least two packet header values, said parse value being indicative of whether the transport and network layer protocols of an associated packet are the TCP and IP protocols.

19.(canceled) A TCP offload engine (TOE) capable of offloading TCP protocol processing tasks from a host, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a memory that simultaneously outputs at least two TCP state values associated with the TCP connection, wherein said at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number, the memory also simultaneously outputting at least two packet header values of a packet communicated over the TCP connection, wherein said at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

means for receiving said at least two TCP state values and said at least two packet header values and for generating therefrom a signal indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied by the memory and to the means simultaneously.

20.(canceled) The TOE of Claim 19, wherein the memory comprises a first memory portion and a second memory portion, the first memory portion storing said at least two TCP state values, the second memory portion storing said at least two packet header values.

21.(canceled) The TOE of Claim 19, wherein the means is also for updating TCP state values by writing TCP state values into the memory.

22.(canceled) The TOE of Claim 19, wherein the means comprises no sequential logic elements, the means consisting entirely of combinatorial logic elements.

23.(canceled) The TOE of Claim 22, wherein the means is a part of a state machine.

24.(canceled) The TOE of Claim 19, wherein the exception condition is a condition that results in control of the TCP connection being passed from the TOE to the host.

25. (New) A TCP offload engine (TOE) capable of performing TCP protocol processing tasks, the TCP protocol processing tasks being associated with a TCP connection, the TOE comprising:

a first memory that stores and simultaneously outputs at least two TCP state values associated with the TCP connection, wherein the at least two TCP state values are taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number;

a second memory that stores and simultaneously outputs at least two packet header values of a packet communicated over the TCP connection,

wherein the at least two packet header values are taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value; and

a state machine that includes an amount of combinatorial logic, wherein the combinatorial logic receives said at least two TCP state values and said at least two packet header values and generates therefrom a signal indicative of whether an exception condition has occurred, the two TCP state values and the two packet header values all being supplied to the combinatorial logic simultaneously, and wherein the state machine causes the expected packet receive sequence number value and the receive packet sequence limit value in the first memory to be updated simultaneously.

26.(New) The TOE of Claim 25, wherein the TOE is capable of offloading the protocol processing tasks from a host processor, wherein the host processor performs exception protocol processing associated with the TCP connection if the exception condition has occurred.

27.(New) The TOE of Claim 26, wherein the first memory, the second memory and the state machine are all parts of a single integrated circuit.

28.(New) The TOE of Claim 27,

wherein the first memory simultaneously outputs at least three TCP state values taken from the group consisting of: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit sequence number,

wherein the second memory simultaneously outputs at least three packet header values taken from the group consisting of: a receive packet sequence number, a packet payload size, a packet acknowledge number, and a packet transmit window value, and

wherein said at least three TCP state values and said at least three packet header values are all supplied to the combinatorial logic simultaneously.

29.(New) The TOE of Claim 27, wherein the single integrated circuit is an integrated circuit of a CPU chip set.

REMARKS

Reconsideration and allowance is respectfully requested.

Claims 2, 5, 10, 12, 13 and 15-18

The Office Action indicates that Claims 2, 5, 10, 13 and 15-18 would be allowable if rewritten into independent form (Office Action, page 5, lines 9-12). It is assumed that Claim 12 should also be in the list because Claim 12 is in the list of "objected to" claims in the Office Action Summary page, is not listed in the list of rejected claims in the Office Action Summary page, and is not rejected over any prior art in the Office Action. Accordingly, Applicants have rewritten Claims 2, 5, 10, 12, 13 and 15-18 into independent form. Allowance of Claims 2, 5, 10, 12, 13 and 15-18 is respectfully requested.

Dependent Claims 3, 4, 6-9 and 14

Dependent Claims 3, 4, 6-9 and 14 have been rewritten to depend upon claims indicated by the Examiner to be allowable if rewritten into independent form. Allowance of dependent Claims 3, 4, 6-9 and 14 is therefore requested.

The §103 Rejection of Independent Claims 1 and 11

Independent Claims 1 and 11 are rejected under 35 U.S.C. §103 over Jolitz (Patent Pub: 2001/0025315) in view of Connery (USP 5,937,169). The Office Action cites Jolitz and states:

"Referring to claim 1, Jolitz teaches: A Network Accelerator (TCP offload engine) per Fig 1 which is connected to a host per Pg 3 Para [0032] and which has registers and ADE 2 (***First memory***) in which expected values associated with TCP/IP headers are stored per Pg 7 Para [0076]. The RX engine in the Network Accelerator stores TCP/IP header values in the Proto memory and RX register (***second memory***) per Pg 7 Para [0077]. The applicant broadly defines

FEB 23 2006

a "flush detect signal indicative of whether an error has occurred". The RX engine (140 per Fig 6) has a header matcher with logic (150 per Fig 6 or Pg 7 Para [0077] which determines if there is a match otherwise the packet is routed to Rx bypass memory or flushed. The reference further teaches in the event that an error is recognized then the operation is suspended which the examiner interprets as a "flush detect signal indicative of whether an error has occurred"." (emphasis added, Office Action, page 2, lines 11-20).

The Examiner, however, does concede that Jolitz "does not expressly call for: two specific values of packet header: such as, packet sequence number or packet acknowledgement number." (Office Action, page 2, lines 21-23).

The Examiner, however, states that Jolitz "teaches that static values of TCP/IP headers are utilized for comparison", and that cites Connery for a teaching that "sequence number" and "acknowledgment number" are two specific values of packet header. Evidently, the Examiner's reasoning is that Jolitz teaches that static packet header values should be used in Jolitz's comparison, and that it would have been obvious to use "sequence number" and "acknowledgment number" as static values because "Connery teaches: that sequence number and acknowledgment number as well as window are standard static values in a TCP/IP header per Fig 4." (Office Action, page 2, lines 24-28).

Applicants respectfully disagree with the §103 rejection of Claims 1 and 11 and traverse as follows.

<u>No Prima Facie Rejection under 35 U.S.C. §103.</u>

As Applicants and the undersigned understand the Office Action, the Examiner maintains: 1) that "registers and ADE 2" in Jolitz are the "first memory" recited in Claim 1; 2) that the "proto memory" and "Rx register" in Jolitz are the "second memory" recited in Claim 1; and 3) that "header matcher" 150 in Jolitz is the "combinatorial logic" recited in Claim 1.

In response, it is respectfully submitted that the §103 rejection is not

FEB 23

complete enough to be a prima facie rejection under 35 U.S.C. §103. The rejection does not say what specific "registers" the Examiner maintains are part of the Claim 1's "first memory". There are many different "registers" mentioned in the Jolitz document. Which one or ones is it that the Examiner maintains is or are part of the "first memory"? Is "prototype register" 148 of Figure 6 one of the "registers" referred to by the Examiner? The Office Action does not say.

Also, the Office Action's reference to "ADE 2" is confusing and incomplete. There is no "ADE 2" in the Jolitz document as far as Applicants can see. Is the Examiner referring to "ADE memory" 56[1] in Figure 3. Is the Examiner referring to "ADE register" 156 in Figure 6? Is the Examiner referring to both? Or is the Examiner referring to something else? What is the "2"? What specific block in what specific figure is it that the Examiner maintains is Claim 1's "second memory"?

Applicants also do not understand the Examiner's reference to "proto memory." Is the Examiner's mention of "proto memory" referring to "proto memory 50" in Figure 3, or to "Proto Mem" 52 in Figure 6, or to "prototype register" 148 in Figure 6, or to some or all of these? Clarification is requested.

Also, the Examiner's reference to "RX register" is confusing. Is the Examiner's mention of "RX register" referring to "TCP Header Rx Register" 152 of Figure 6, or to "IP Header Register" 154 of Figure 6, or both? Applicants note that register 154 is an "IP" header register, and none of the TCP state values or packet header values of Claim 1 are IP values. Or is the Examiner referring to "Rx Window Register" 170 of Figure 6? Clarification of the rejection is requested.

Applicants need to know specifically what the Examiner is identifying in Jolitz as being the claimed "first memory," and specifically what the Examiner is identifying in Jolitz as being the claimed "second memory" so Applicants can examine whether outputs from these two memories are supplied "simultaneously" to "header matcher" 150. As it stands now, Applicants cannot decipher

specifically what the Examiner is maintaining the "first memory" is, and specifically what the Examiner is maintaining the "second memory" is. The rejection is not a prima facie §103 rejection.

Claim 1: Jolitz Does Not Disclose Simultaneously Supplying Two TCP State Values and two Packet Header Values To "Header Matcher" 150

Claim 1, very importantly, recites that "the two TCP state values and the two packet header values all being ***supplied to the combinatorial logic simultaneously***" (emphasis added). There is no such teaching or disclosure in Jolitz.

The Examiner's attention is directed to Figure 6 of Jolitz. How does the Examiner know[2] that "TCP/IP header matcher" block 150 "simultaneously" receives information from blocks 152 and 154 and 156 and 148? Jolitz Figure 6 is a non-enabling confusing rat's nest of tersely labeled boxes and unlabeled arrows. The functionalities of the boxes are only vaguely alluded to; there are few specifics. What the arrows are is also unknown. Are the arrows control? Are they data? Are they both? What specific information is being carried? There are no signal names on the arrows at all. For example, there is a dark, double-arrowed line labeled "Rx Control Bus" that extends from block 140 to the left across the page and then down to the lower left corner of the page. The line also extends vertically down to the right of the left-most column of boxes 164, 152 and 154. The "TCP Header Rx Register" box 152, the "IP Header Rx Register" box 154, and the "ADE register" box 156 are drawn coupled to this bus. But what is the width of the connection between box 152 and the bus? It is not shown. What is the width of the connection between box 154 and the bus? It is

---

[1] "ADE memory" 56 in Figure 3 is not shown to be in communication with "Rx Engine" 48.
[2] For the discussion here, it is assumed that "ADE register" 156 and "prototype register" 148 in Figure 6 are what the Examiner calls the "first memory"; that blocks 152 and 154 of Figure 6 are what the Examiner calls the "second memory", and that block 150 is what the Examiner calls the "combinatorial logic".

not shown. What is the width of the connection between box 156 and the bus? It is not shown. Can "TCP/IP header matcher" 150 simultaneously receive outputs from block 152 and from block 154 and from block 156? The text of the Jolitz document does not say.

Figure 6 and the corresponding text in [0075]-[0079] is exceedingly confusing and incomplete, but the diagram of Figure 6 would probably have led one to believe that the three boxes 152, 154 and 156 do **NOT** simultaneously output to box 150. In Figure 6, there is one arrow extending to the right out of box 152 to the bus, and another arrow extending to the right out of box 154 to the bus, and another arrow extending to the left out of box 156 to the bus, and yet there is only one arrow extending to the right from the bus into box 150. Presumably the header matcher 150, under some kind of control from "Rx Engine ***control*** state machine" 140, receives from each of the three boxes 152, 154 and 156 sequentially, one at a time, across the bus in common fashion and into the single arrow that leads into box 150. Applicants submit that there is no statement in Jolitz that states that TCP state values from ADE register 156 and "prototype register" 148 are supplied to block 150 at the same time that packet header values from registers 152 and 154 are supplied to block 150. The values could be received sequentially.

Further evidence that the Jolitz device of Figure 6 is probably doing sequential processing in making the decision of whether to route the segment to the bypass memory (see the first sentence of paragraph [0079]) is found in the part of the text from the last sentence of [0077] to the end of the first sentence of [0079]. This portion of text describes state machine 140 operating in concert with ALU 164 to do updating of Rx register 152, to do window size adjustment, to reduce the value in window register 170 as data is received, and to increase the value in window register 170. After explaining that ALU 164 is involved in these operations and that ALU 164 is under the control of state machine 140, Jolitz says "when processing a packet header, if any of the fields of the header do not

match expected values, the segment may be routed to the Rx bypass memory 62, and the Rx engine may go into the idle state." (emphasis added, '315, [0079]). From the context here, and from the lack of mention of matcher box 150, it would appear to one of ordinary skill that ALU makes the determination of whether there is a "match" to the "expected values". ***A common function of an ALU is to compare two values to determine if they "match".*** If ALU 164 is performing the "match" determination, then Applicants would presume that either matcher 150 is not involved in the determination or that both ALU 164 and matcher 150 are involved in the determination. The discussion of ALU 164 in this section of text coupled with the failure to mention matcher 150 casts doubt on the Examiner's assumption[3] that matcher 150 alone performs all the comparing or matching involved in making the decision of whether to route the segment to the bypass memory.

Applicants and the undersigned have spent hours and hours trying to understand Figures 3 and 6 and paragraphs [0075]-[0079], and in particular the vague statement in paragraph [0077] "A TCP/IP header matcher 150 which contains logic for comparing of session fields of the packet obtained from the prototype register and variable fields held in a TCP header Rx register 152 and IP header Rx register 154." (This is not even a sentence). After spending all that time, Applicants have come to the conclusion that: 1) it cannot be determined whether two of the listed "TCP state values[4]" of Claim 1 are ever supplied to Jolitz's "Header Matcher" 150, 2) Jolitz nowhere either discloses or suggests that "header matcher" 150 receives values from boxes 152, 154, 156 and 148 simultaneously, and 3) it cannot be determined precisely what the alluded to "comparing" really entails. Reconsideration by the Examiner is respectfully

---

[3] The Office Action states "The RX engine (140 per Fig 6) has a header matcher with logic (150 per Fig 6 or Pg 7 Para [0077] which determines if there is a match otherwise the packet is routed to Rx bypass memory or flushed" (Office Action, page 2, lines 16-18).

[4] The two "TCP state values" must be taken from the group: "Receive packet sequence limit", "expected receive packet sequence number", "transmit sequence limit", "a transmit acknowledge number", and "a transmit sequence number".

requested.

If the Examiner sustains the §103 rejection, then the Examiner is respectfully requested to state on the record how he knows that two of the specifically listed TCP state values[5] are transferred to header matcher block 150. If the Examiner sustains the §103 rejection, then the Examiner is respectfully requested to state on the record how he knows that values from boxes 152, 154, 156 and 148 are all simultaneously supplied to header matcher box 150. Applicants respectfully submit that there is no such teaching or disclosure in Jolitz.

For all these reasons, it is submitted that the §103 rejection of independent Claim 1 is improper and should be withdrawn.

<u>Claim 11: Jolitz Does Not Disclose Generating a Signal Indicative of an Exception Condition Within Approximately One Clock Period As Claimed</u>

Claim 11, very importantly, recites "the hardware state machine is ***clocked by a clock signal***, wherein the hardware state machine generates said signal from said at least two TCP state values and said at least two packet header values ***within approximately one period of the clock signal***" (emphasis added). There is no such teaching or disclosure in Jolitz.

The Office Action states that "The RX Engine (140 per Fig 6)(hardware state machine) has a header matcher with logic (150 per Fig 6 or Pg 7 Para[0077] which determines if there is a match otherwise the packet is routed to Rx bypass memory or in the event that an error is recognized then the operation is suspended which the examiner interprets as a 'signal indicative of whether an exception condition has occurred.' Jolitz teaches that the output of the accelerator runs at the same clock rate as the signaling per Pg 3 Para [0029] or

FEB 23 2006

wherein the hardware state machine is clocked by a clock signal and output a decision within approximately one period of the clock signal." (Office Action, page 3, lines 21-28).

Applicants respectfully submit that the Examiner is doing a bit of handwaiving here. Where is the clock signal? What paragraph [0029] says is that there is "a mechanism that processes the costly portions of standard protocols in hardware entirely, and to do so at the same clock rate of the signaling." ('315, [0029]). But what does that mean? That statement in [0029] does not preclude state machine 140 of Figure 6 from being clocked many times during the receipt of a single packet. And if the state machine is clocked many times, then logic in "header matcher" could perform parts of a larger logic equation sequentially such that the results are later combined to obtain a single signal that is "indicative of whether an exception condition has occurred". Applicants respectfully request that the Examiner point out where the "clock signal" is that the Examiner says clocks state machine 140. Only once that clock signal has been identified can it be determined whether the header matcher 150 outputs the recited signal in approximately one period of the clock signal. Applicants respectfully submit that Jolitz does nothing to teach or disclose any relationship between the clocking of state machine 140 and the operation of header matcher 150. There is very little information on what the Rx Engine Control State Machine 140 does, and almost no information on how it operates. As set forth above with respect to Claim 1, it is possible that state machine 140 sequentially loads information from other boxes (for example, boxes 156, 152 and 154) into header matcher box 150 across the "Rx Control Bus", and that this occurs in response to multiple clock edges of the clock signal that clocks the state machine. There is no statement to the contrary in the Jolitz document. Reconsideration and withdrawal of the improper §103 rejection is respectfully

---

[5]The listed TCP state values are: a receive packet sequence limit, an expected receive packet sequence number, a transmit sequence limit, a transmit acknowledge number, and a transmit

requested.

Connery's Sequence Number and Acknowledgement Number In the TCP Header Template 112 Are Not "Static"

Perhaps this is a minor point, but in Applicants' opinions the sequence number and the acknowledgement number in Connery's TCP header template 110 in Figure 4 are not "static." Applicants submit that the Examiner has made a mistake. In explaining the "sequence numbers", Connery states "The sequence number in the template is used in the header for the packet carrying the first segment of the datagram. Sequence numbers are ***updated*** automatically by the smart interface card for each subsequent packet carrying a segment of the datagram." (emphasis added, '169, col. 12, lines 2-5). It is submitted that the sequence number is updated and is not "static." Next, in explaining the "acknowledgement number" in the TCP header template, Connery states "The next field in the TCP header template is the acknowledgment number. This is a 32 bit control field that contains the value of the ***next sequence number*** a sender of the segment is expecting to receive if the ACK bit is set. ...Thus, the value is included in the TCP header template and copied directly for each packet of the plurality of packets sent....In other embodiments, the acknowledgement number is automatically ***updated*** using processing in the smart interface during the processing in the smart interface during the processing of the datagrams." (emphasis added, '169, col. 12, lines 7-17). Applicants therefore submit that Connery does not teach the "sequence number" and the "acknowledgement number" in the TCP header template are "static" values. As such, one of ordinary skill would not have been led by Connery to modify Jolitz's "comparison" so as to use the sequence number and the acknowledgement number in the comparison as the Examiner proposes.

---

sequence number.

Jolitz '315 Is Not Enabling, and Therefore Is Not A Proper Reference

The Jolitz published patent application is incomplete, confusing and vague in important places. It is exceedingly poorly organized and written. It is very difficult to understand, even for an expert in this field with the benefit of many years of having designed working TCP Offload Engine hardware. Considering the relatively low level of ordinary skill in this art in December 2003[6], it is not appropriate to look at the Jolitz disclosure with the benefit of already knowing how to make Applicants' claimed invention. It is not appropriate to expect one of ordinary skill, who was trying to learn how to make a TCP offload engine, to spend hours and hours of time trying to understand vague statements and reconcile them with incomplete diagrams. The impossibility of determining how information is supplied to header matcher block 150 (see the discussion above) is just one example. The impossibility of determining specifically what that information is is another example. The impossibility of determining what the checking or "matching" equation is is another example. Applicants submit that one of ordinary skill as of December 2003 could not take the Jolitz disclosure and be taught by the Jolitz document now to build a TCP offload engine that worked without undue experimentation. Jolitz, frankly, would confuse one of ordinary skill to the point that one of ordinary skill likely would stop trying to figure it out and would set off to design a TCP offload engine from scratch or using other teachings (such as Alacritech's patents). Because the Jolitz document is so confusing and vague and incomplete, it would not have enabled one of ordinary skill to make the structure of Figure 6 work, and as such is not a proper reference in the Jolitz/Connery §103 combination.

---

[6] As of December 2003, there was only one TOE NIC (TCP Offload Network Interface Card) on the market that could receive control of a TCP connection from a host and pass control of a TCP connection to a host, and that TOE NIC was made by Alacritech, assignee of the present patent application. It is submitted that as of December 2003, one of ordinary skill in this field would not have had any real-world working knowledge of designing and building TOE NICs of this type that worked.

Claims 19-24

Claims 19-24 are canceled without prejudice. Applicants reserve the right to resubmit these claims at a later time in a continuation.

New Claims 25-29

Claim 25 (and dependent Claims 26-29) recites: 1) combinatorial logic that simultaneously receives at least two specifically listed TCP state values and at least two specifically listed packet header values and generates therefrom a signal indicative of whether an exception condition has occurred, and further recites that 2) simultaneously updating the expected packet receive sequence number value and the receive packet sequence limit value in the first memory. ***Jolitz provides an enabling disclosure of neither*** novel aspect. Jolitz certainly does not disclose simultaneously updating both an expected packet receive sequence number value and a receive packet sequence limit value in the first memory as claimed in Claim 25. Connery does nothing to remedy the shortcoming in the Jolitz document. Applicants therefore submit that the subject matter of Claims 25-29 is patentably distinct over Jolitz and/or Connery.

Conclusion

In view of the foregoing amendments and remarks, Applicants respectfully submit that the present application (Claims 1-18 and 25-29 are pending) is in condition for allowance. If the Examiner would like to discuss any aspect of this application, the Examiner is requested to contact the undersigned at (925) 621-2115.

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as First Class Mail on the date indicated below and is addressed to:

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450.

By Lester Wallace
T. Lester Wallace

Date of Deposit: August 29, 2005

Respectfully submitted,

Lester Wallace

T. Lester Wallace
Attorney for Applicants
Reg. No. 34,748

Silicon Edge Law Group LLP
6601 Koll Center Parkway
Suite 245
Pleasanton, CA 94566
(925) 621-2115